TP6 : Manipulation d'images

Dans ce TP, vous apprendrez à lire des images PNG en C (en s'aidant d'une librairie extérieure) et ferez une quelques opérations sur les images.

1 Le TP5 : dernière chance pour le corriger et le compléter (40mn maximum)

Vous trouverez, sur Moodle, le résultat des évaluations automatiques du TP5, sur le décodage de Vigenère. Si votre programme a été noté comme ayant des problèmes, vous pouvez le corriger dès à présent, et effectuer un nouveau dépôt de votre travail (ce sera le dernier concernant le TP5). N'oubliez pas de suivre à la lettre les instructions de rendu données dans le TP5. Vous devrez effectuer votre rendu avant 9h10.

2 Lire des images en C

Le C ne possède pas, nativement, de fonction permettant de gérer facilement les images. Il est tout à fait possible d'utiliser la fonction **fopen** puis de réaliser sa propre routine de lecture, mais une image est rarement écrite telle quelle sur le disque : elle est souvent compressée (JPEG, PNG, ...), et il faut alors implémenter une routine de décompression pour pouvoir la lire.

Pour éviter d'avoir à faire cela, nous allons utiliser une librairie extérieure de lecture d'images appelée LodePNG (`http://lodev.org/lodepng/`). Les fichiers *lodepng.c*, *lodepng.h*, *example_ decode.c* et *example_ encode.c* sont inclus parmi les fichiers de ce TP.

Lorsque vous utilisez LodePNG pour ouvrir une image, vous obtiendrez un tableau de **unsigned char** de type RVBa :

- La première case du tableau indique la quantité de rouge du premier pixel de l'image (0 pour signifier que le rouge est absent, 255 pour signifier qu'il est très intense),
  item La seconde case du tableau indique la quantité de vert du premier pixel de l'image (0 pour signifier que le vert est absent, 255 pour signifier qu'il est très intense),
  item La troisième case du tableau indique la quantité de bleu du premier pixel de l'image (0 pour signifier que le bleu est absent, 255 pour signifier qu'il est très intense),
- item La quatrième case du tableau indique la transparence du premier pixel de l'image (255 pour signifier que le pixel est complètement opaque, 0 pour signifier qu'il est complètement transparent),
- La cinquième case du tableau indique la quantité de rouge du second pixel de l'image,
- Etc...

Questions

1. En vous inspirant des fichiers *example_ decode.c* et *example_ encode.c*, écrivez un programme qui ouvre une image PNG (vous pouvez en trouver parmi les fichiers joints au TP) et l'écrit dans un autre fichier.
   Pour compiler un fichier utilisant la librairie LodePNG, il vous faudra inclure le fichier *lodepng.h* dans votre programme principal :

```
#include "lodepng.h"
```

   A la compilation, vous devrez compiler votre programme en même temps que le fichier *lodepng.c* :

```
gcc mon_prog.c lodepng.c -o mon_prog.exe
```

Assurez-vous que les fichiers *lodepng.c* et *lodepng.h* sont dans le même répertoire que votre fichier programme.

2. Pour rendre la suite des opérations plus simple, vous devrez stocker l'image dans une structure de données appelée **myimage**, qui contiendra 6 champs : la hauteur de l'image, la largeur de l'image, et quatre tableaux de **unsigned char**, qui seront le canal rouge, vert, bleu et alpha de l'image. Proposez une telle structure.

3. Proposez une fonction **myimage LireImage(char* nom_fichier)** permettant de lire une image PNG et la stocker dans une structure **myimage**.

4. Proposez une fonction **void EcrireImage(myimage im, char* nom_fichier)** permettant d'écrire une **myimage** dans le fichier **nom_fichier**.

3 Transformer une image en noir et blanc

Dans cette partie, vous transformerez une image couleur en image noir et blanc. Une image noir et blanc possède la particularité que, pour chaque pixel, la quantité de rouge, de vert de de bleu est la même. Pour convertir une image couleur en noir et blanc, on remplace simplement, pour chaque pixel, la valeur de rouge, de vert et de bleu par la moyenne de ces trois valeurs. Il est important que, dans le fichier rendu, vous ne copiez/collez pas le code de la librairie lodepng, mais que vous l'incluiez dans votre programme.

Questions

Proposez une fonction **void NoirEtBlanc(myimage im)** qui transforme l'image passée en paramètre en noir et blanc, et testez la.

Attention lorsque vous calculerez la moyenne des pixels au dépassement de capacité lors de l'addition.

4 Rendu de votre programme

Vous devez rendre le travail réalisé pour le TP6 sur Moodle (partie image noir et blanc).
Ce rendu doit suivre un format spécifique. Votre programme devra s'exécuter à l'aide de la commande

```
./noir_et_blanc entree.png sortie.png
```

— **entree.png** doit être un fichier PNG couleur, qui sera transformé en noir et blanc par votre programme.

— **sortie.png** sera le fichier PNG de sortie de votre programme, à savoir l'image d'entrée convertie en noir et blanc.

— Si votre programme ne parvient pas à se terminer correctement (fichier d'entrée introuvable, mauvais nombre de paramètres en entrée du programme, etc), il devra renvoyer **EXIT_FAILURE**. Sinon, il devra se terminer avec le code **EXIT_SUCCESS**.

Votre programme devra compiler à l'aide de la commande

```
gcc noir_et_blanc.c lodepng.c -o noir_et_blanc
```

Si votre ligne de compilation est plus complexe, vous devrez alors la spécifier dans un *Makefile*. Votre ou vos fichiers source (sauf les les fichiers lodepng.c et lodepng.h - inutile de rendre ces fichiers) devront être directement placés dans un fichier zip (et ne pas être dans un sous dossier du fichier zip), dont le nom sera

```
VotreNom_VotrePrenom_VotreNumeroEtudiant.zip
```

sans aucun espace (si votre nom ou votre prénom contiennent un espace, ne les faîtes pas figurer). Les formats de compression acceptés sont .zip, .tgz, .tar.gz, .7z, .rar.

Si vous avez un binôme, il vous faudra écrire son nom, prénom et numéro d'étudiant séparés par des espaces, en plus des vôtres, dans un fichier *binome.txt*, dont le format sera

```
Nom1 Prenom1 Numero_etudiant1
Nom2 Prenom2 Numero_etudiant2
```

Vous ne devez rendre le fichier qu'une seule fois par binôme.